

Analog Devices CCES Debugger

Abstract

This document describes the setup and handling for the CrossCore Embedded Studio debugger/simulator as target system.

Please note: *The CrossCore Embedded Studio debugger/simulator does not support debugging features when executing tests with TESSY. The normal debugger GUI is not accessible during the TESSY test execution.*

Table of Contents

Abstract	1
1 Introduction.....	2
2 Setup.....	2
2.1 Windows environment variable CCES_HOME	2
2.2 CCES session configuration.....	2
2.3 TESSY Environment Editor (TEE) Settings.....	4
2.3.1 Char Size attribute for the SHARC.....	4
2.3.2 Adjust the target settings.....	5
3 Executing a Test Run	8
4 Interactive Debugging.....	8
4.1 Step 1: Generate the test driver executable	8
4.2 Step 2: Load the test driver executable into the debugger	9
4.3 Step 3: Set a breakpoint at your test object.....	12
4.4 Step 4: Run until the test object.....	13
5 Known Issues	14
5.1 Connection problem or tthd: Java processes died unexpectedly	14
5.2 Executing test cases separately	14

1 Introduction

TESSY supports the CrossCore Embedded Studio (CCES) debugger/simulator for automatic test execution using an interface DLL provided by Analog Devices. During test execution, there is no interactive debugging possible, i.e. the “Define Breakpoint” feature of the TESSY execution dialog has no functionality for this target environment.

2 Setup

Running tests with the CrossCore Embedded Studio from TESSY requires the installation of CrossCore Embedded Studio software with the minimum versions listed below.

Controller	Required CrossCore Embedded Studio Version
BF518	1.0.3
ADSP-21368, ADSP-SC589	2.0.0

2.1 Windows environment variable CCES_HOME

The use of the interface DLL requires to add the windows environment variable "CCES_HOME". The value of this variable has to be set to the path of your "Analog Devices CrossCore Embedded Studio" installation, e.g. "C:\Program Files\Analog Devices\CrossCore Embedded Studio 2.1.0".

2.2 CCES session configuration

Furthermore you have to know the session configuration consisting of processor, target type, and target platform as it is used within CrossCore Embedded Studio. The session configuration can be found in the Run/Debug Settings of the launch configuration settings.

The following session configurations can be used within TESSY for the Blackfin:

Target	Target type	Platform	Processor
Simulator	ADSP-BF5xx Blackfin Family Simulators	ADSP-BF5xx Single Processor Simulator	ADSP-BF518
Debugger	Blackfin Emulators/EZ-KIT Lites	ADSP-BF518 EZ-KIT Lite via Debug Agent	ADSP-BF518

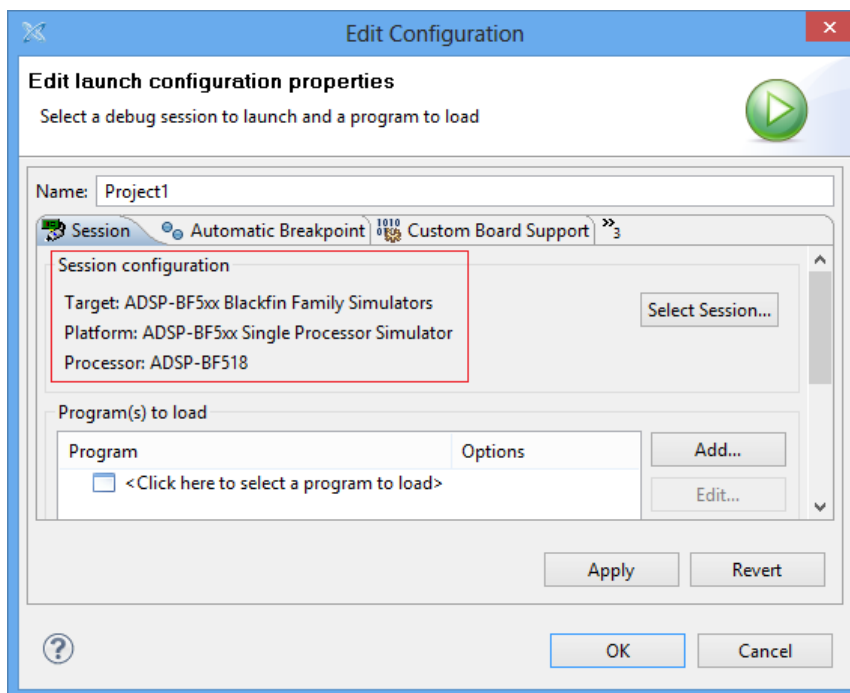
The following session configurations can be used within TESSY for the SHARC:

Target	Target type	Platform	Processor
Simulator	ADSP-2136x Family Simulator	ADSP-2136x Simulator	ADSP-21368
Debugger	Emulation Debug Target	ADSP-SC589 via ICE-2000	ADSP-SC589

The following session configurations can be used within TESSY for the ARM:

Target	Target type	Platform	Processor
Debugger	Emulation Debug Target	ADSP-SC589 via ICE-2000	ADSP-SC589

The predefined setup for the Blackfin compiler within TESSY is prepared for the BF518 Blackfin simulator. The predefined setup for the SHARC compiler within TESSY is prepared for the ADSP-21368 SHARC simulator. The predefined setup for the ARM compiler within TESSY is prepared for the ARM core of the ADSP-SC589. If you are using the debugger or another controller you need to adapt the TESSY environment settings and the makefile template.



2.3 TESSY Environment Editor (TEE) Settings

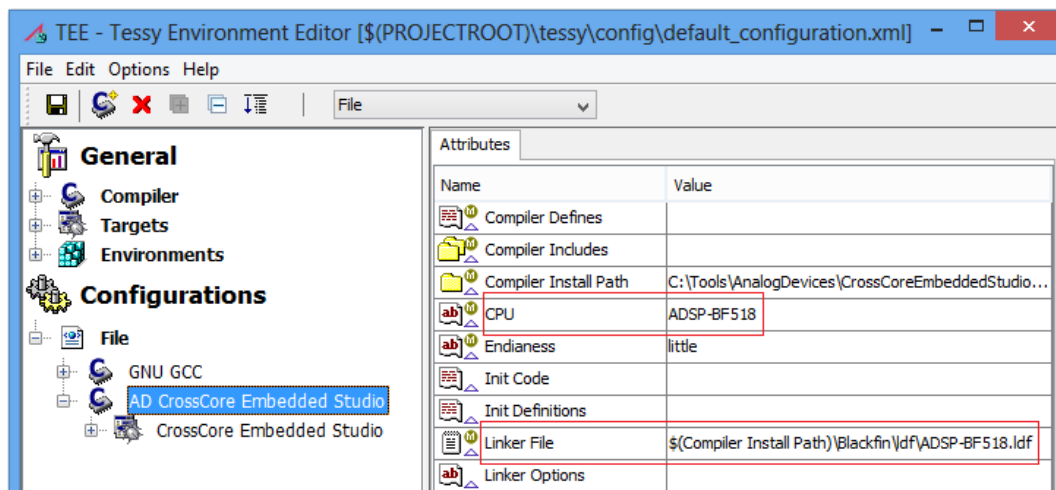
Please consult chapter 6.5 *TEE: Configuring the test environment* from TESSY's user manual if you have not done by now. The TESSY environment editor (TEE) provides some settings that may need to be adapted for the controller you are using.

The following environments need to be enabled for the corresponding target processor families:

Target Processor Family	Target environment to enable
Blackfin	AD CrossCore Embedded Studio
SHARC	AD CrossCore Embedded Studio SHARC
ARM	GNU Tools for ARM

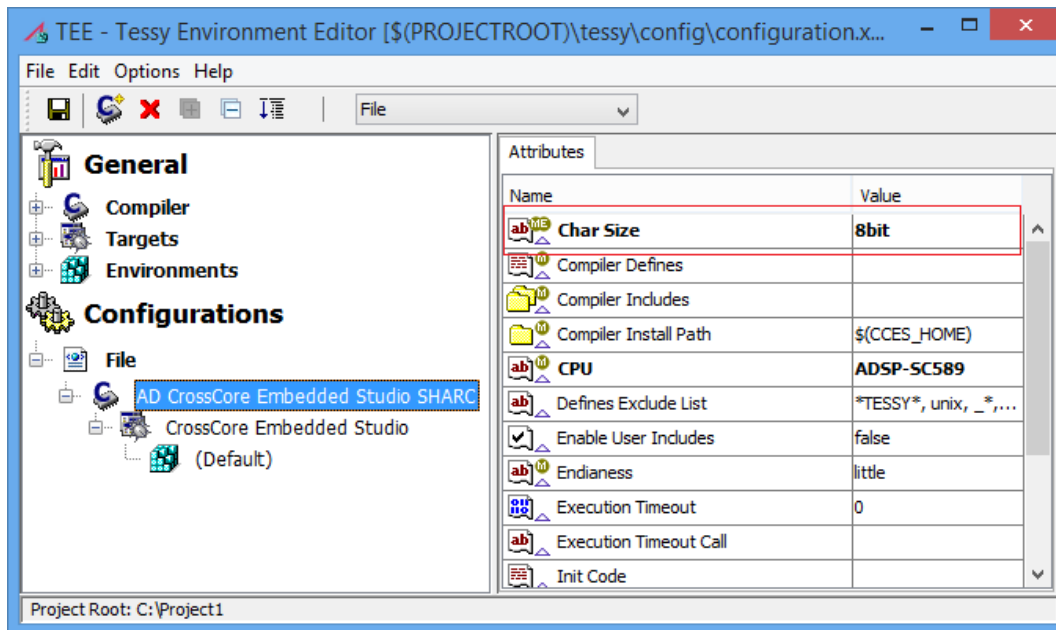
Review the settings within the “AD CrossCore Embedded Studio” compiler node. If you are using another controller than the predefined, you need to change the “CPU” entry according to the list of controller names within the CrossCore Embedded Studio **Session configuration**.

If you need other settings for the linker, you can copy the default linker file into your project location and select this modified file as “Linker File” attribute.



2.3.1 Char Size attribute for the SHARC

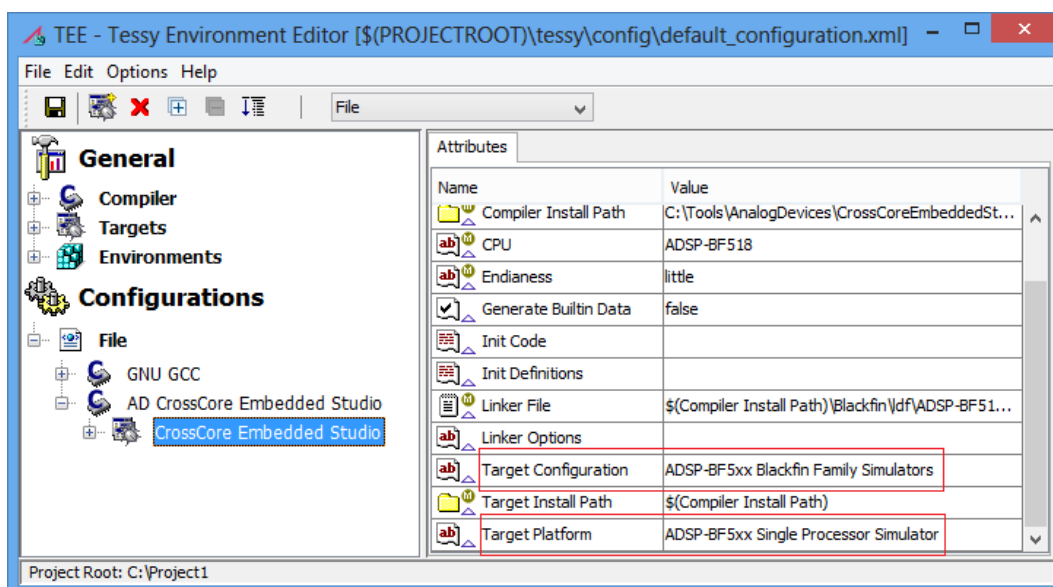
If you are using the “AD CrossCore Embedded Studio SHARC” compiler, you need to check if your CPU supports the character size of 8 bit or 32 bit. Depending on this, you need to set the “**Char Size**” entry either to “8bit” or to “32bit”.



2.3.2 Adjust the target settings

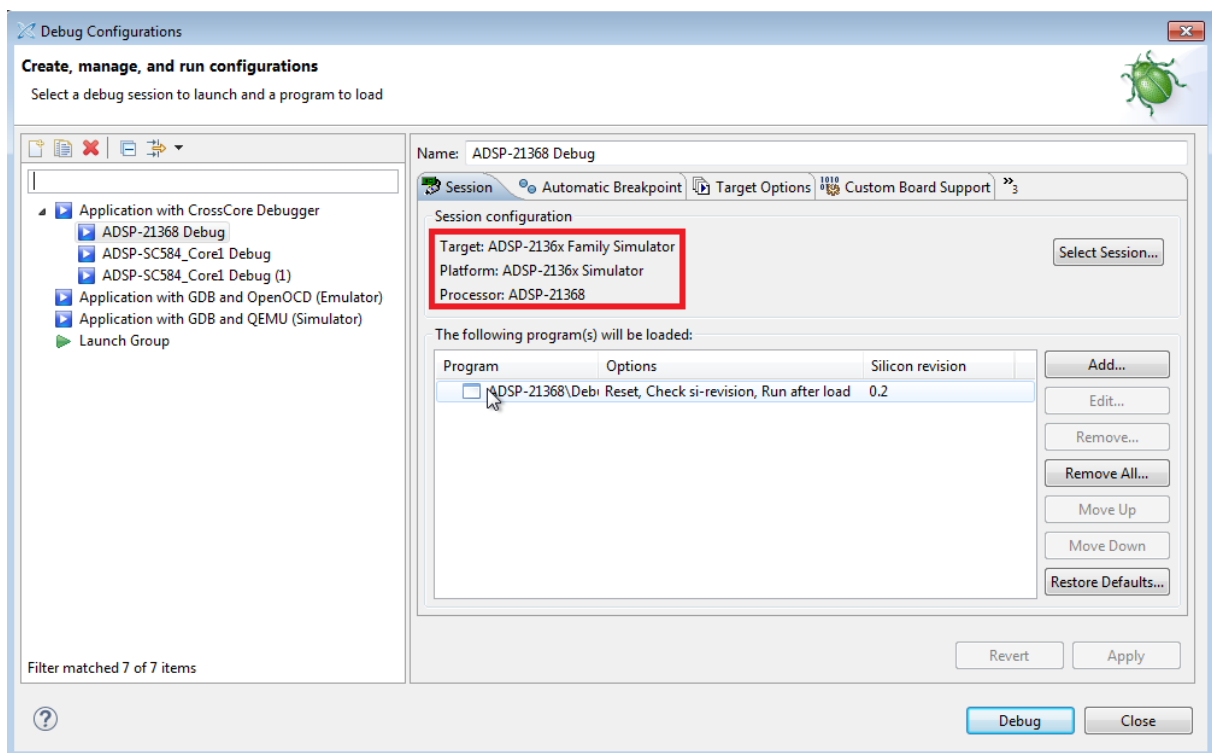
Now select the “CrossCore Embedded Studio” target node and review the attribute settings. The default “Target Configuration” and the "Target Platform" is prepared for the following Targets:




Target Processor Family	Default Target
Blackfin	BF518 Simulator
SHARC	ADSP-21368 Simulator
ARM	ADSP-SC589 Debugger via ICE-2000



If you are using the CrossCore Embedded Studio debugger for a Blackfin target, you need to change the TEE attribute **Target Configuration** to **Blackfin Emulators/EZ-KIT Lites** and the TEE attribute **Target Platform** to **ADSP-BF518 EZ-KIT Lite** via **Debug Agent**.

If you are using another target than the predefined you need to change the **Target Configuration** and the **Target Platform** values according to the CrossCore Embedded Studio session configuration, i.e. the TEE attribute **Target Configuration** is exactly set to the value of **Target:** from CrossCore's **Debug Configurations** dialog as shown below. The value of **Platform:** goes into TEE attribute **Target Platform**, **Processor:** should match TEE's **CPU** attribute.

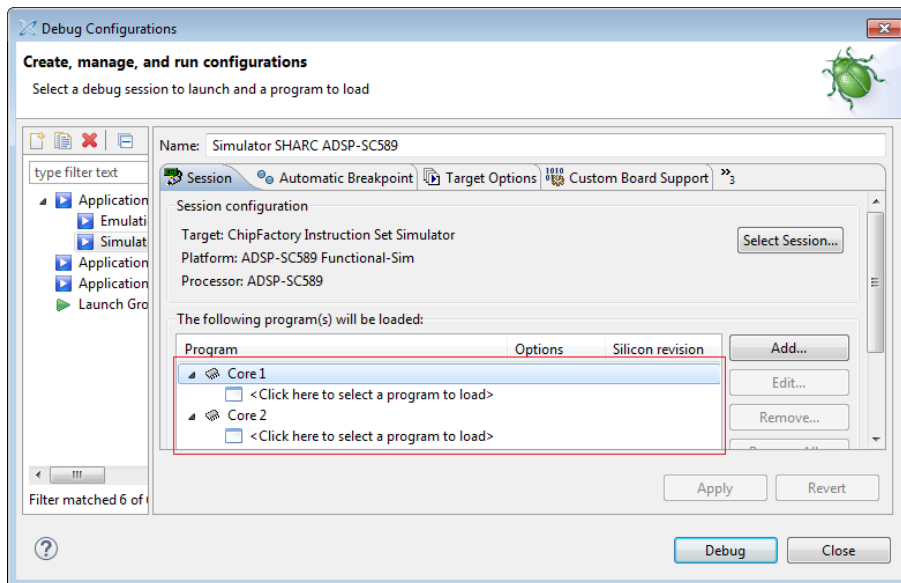


 CPU	ADSP-21368
 Target Configuration	ADSP-2136x Family Simulator
 Target Platform	ADSP-2136x Simulator

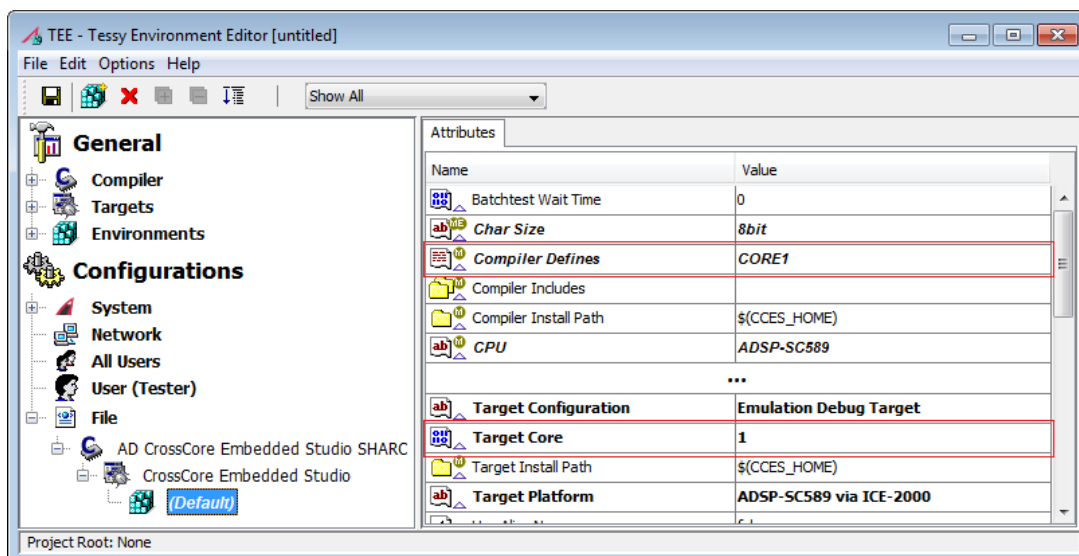
Target Core for Blackfin and SHARC

The TEE attribute **Target Core** determines which target core will be used for multicore targets, e.g. like the ADSP-BF608 or the ADSP-SC589.

The selectable cores of a multicore target can be checked in the Debugger Configuration of CrossCore Embedded Studio.



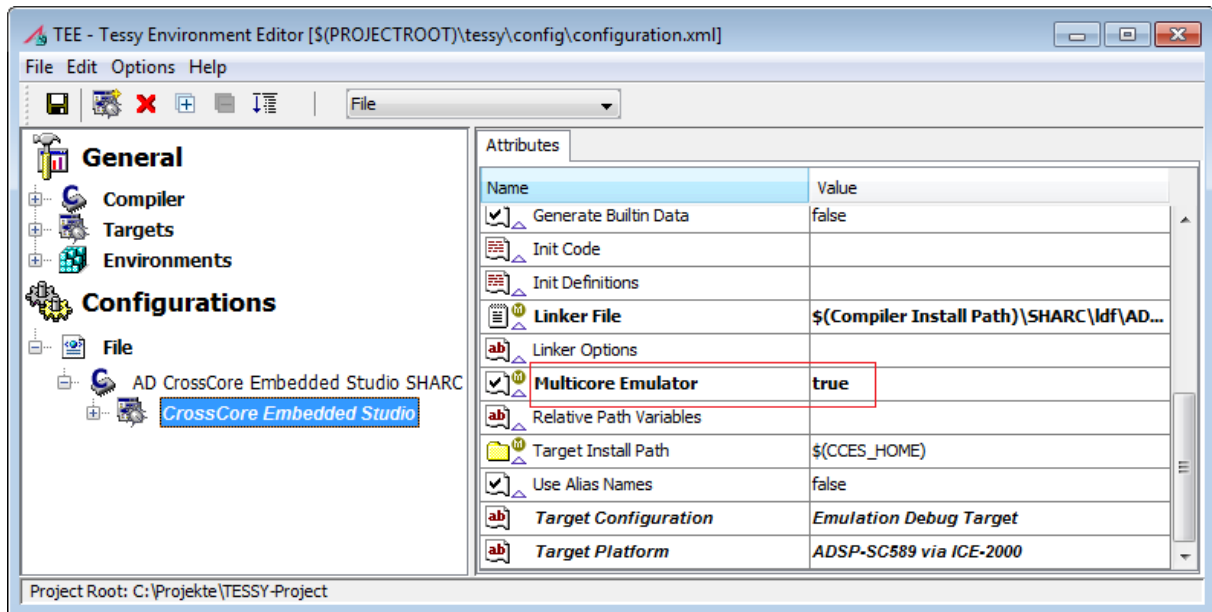
If you are using another core than core 0 you need to change TEE attribute **Target Core** and add the required define to TEE attribute **Compiler Defines**, e.g. **CORE1** as shown below.



Please note: The default setting is core 0. So for an ADSP-SC5xx debugger target the target core attribute needs always to be adapted because the selectable SHARC core IDs are 1 and 2.

SHARC multicore

If you are using the “AD CrossCore Embedded Studio SHARC” compiler and your target CPU is a multicore including an ARM core (e.g. ADSP-SC5xx), which you want to connect via emulator, you need to change the TEE attribute **Multicore Emulator** to **true**. Then the preload to enable core 1 is done automatically.



3 Executing a Test Run

Now you can execute tests by clicking on the **Execute Test** button from TESSY's toolbar menu. Choose the instrumentation options and run a test.

Please note: TESSY will automatically start the CrossCore Embedded Studio debugger in the background and automatically close it when the test is done.

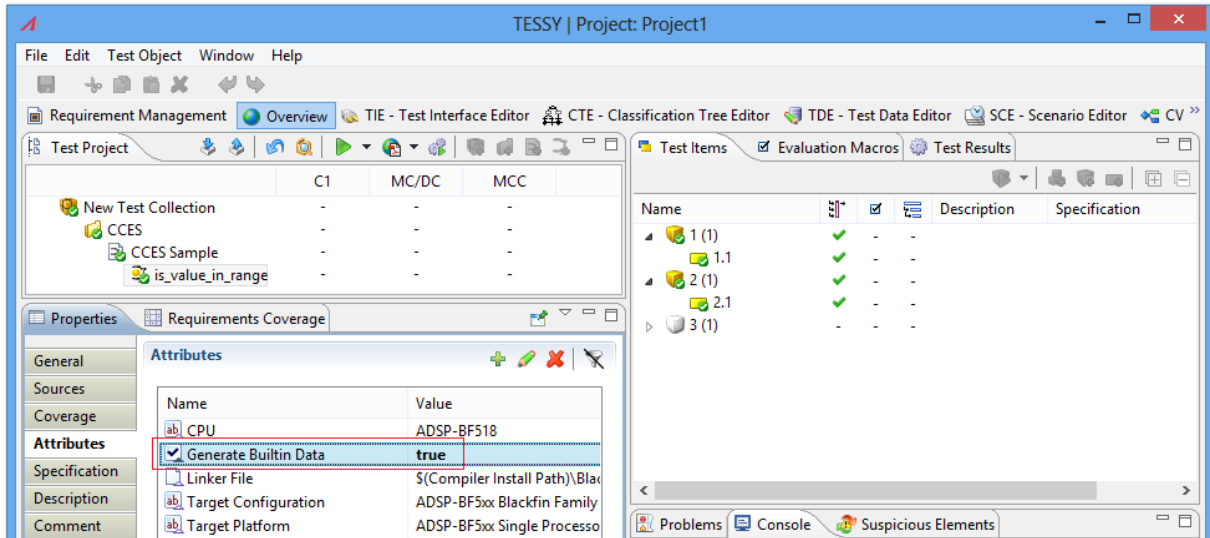
If there are errors within your test results, you may want to debug the test cases. Because the CrossCore Embedded Studio IDE cannot be opened while TESSY controls the execution of the test run you need to follow the steps described below to create a test driver binary containing the test data built-in.

4 Interactive Debugging

The following steps are necessary to create a test driver executable containing the test data built-in that is normally transferred from TESSY to the target debugger automatically during the test run. The executable can be loaded and executed manually into the CrossCore Embedded Studio debugger.

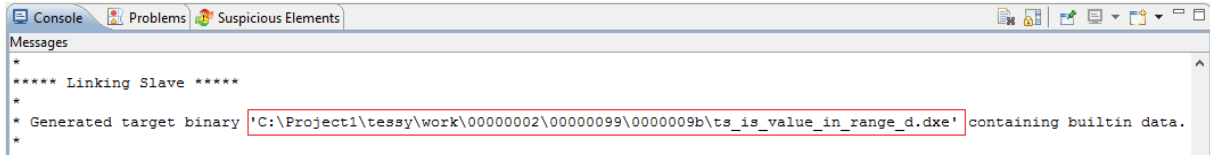
4.1 Step 1: Generate the test driver executable

In order to create a test driver with built-in test data, you need to select the test object and temporarily set the value of TEE attribute **Generate Builtin Data** to **true**.



Now execute the test for the selected test object or test case. It is recommended to set instrumentation to **None** for this purpose in order to debug the original source code without code instrumentation.

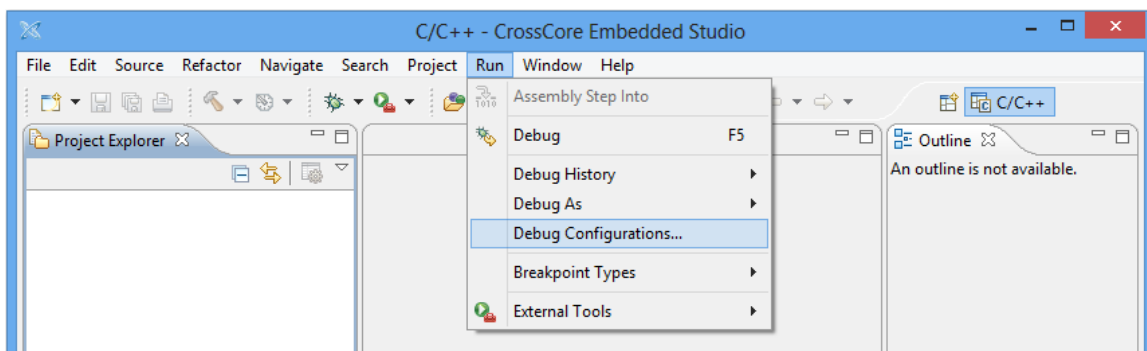
The test driver will be build, the execution will be started in a special mode, and the test driver containing the test data built-in will be built finally. You will see the result of the build process within TESSY's **Console** view:



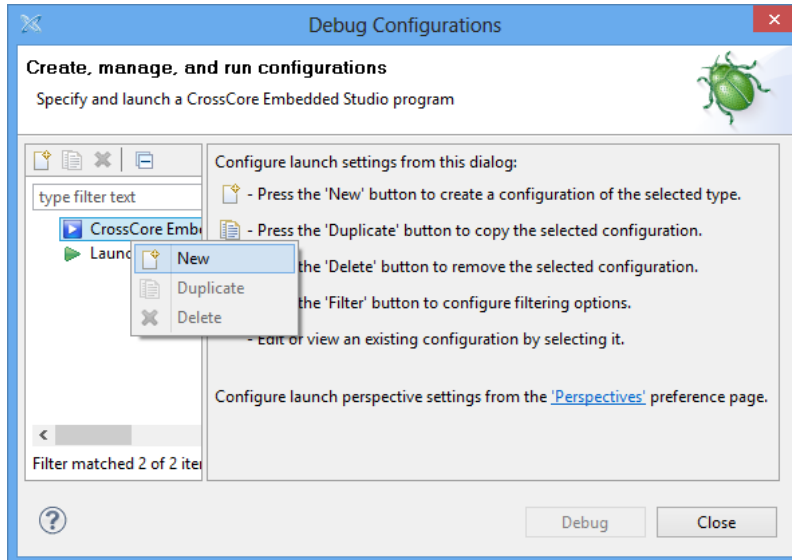
Please note: You need to reset the attribute **Generate Builtin Data** to the default value (**false**) after debugging to switch back the automatic test execution.

4.2 Step 2: Load the test driver executable into the debugger

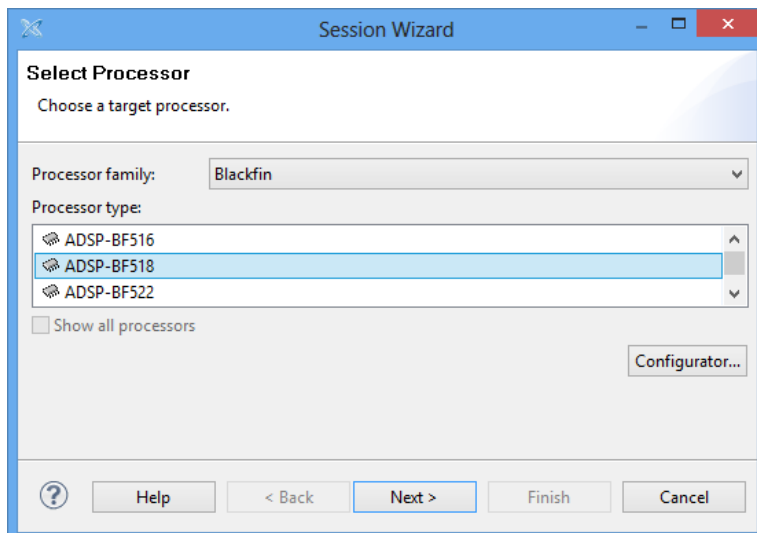
Start the CrossCore Embedded Studio, open the **Run** menu and choose **Debug Configurations....**

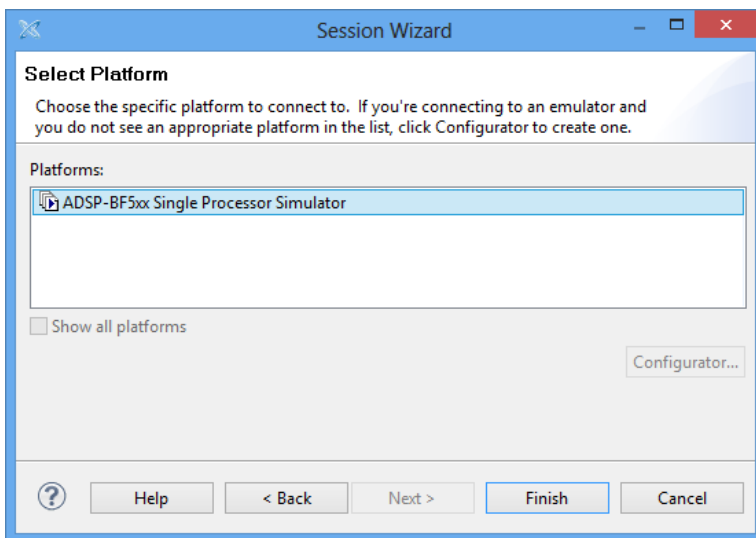
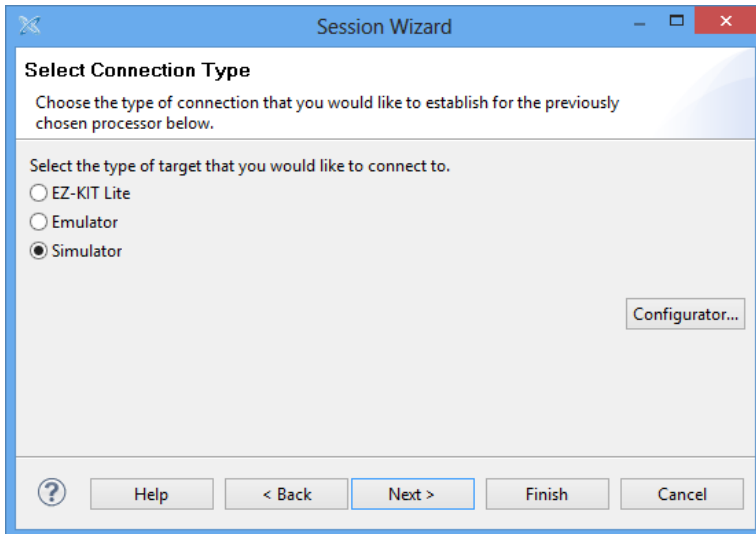


Select **CrossCore Embedded Studio Application** and then add a new configuration by selecting **New** in the context menu:

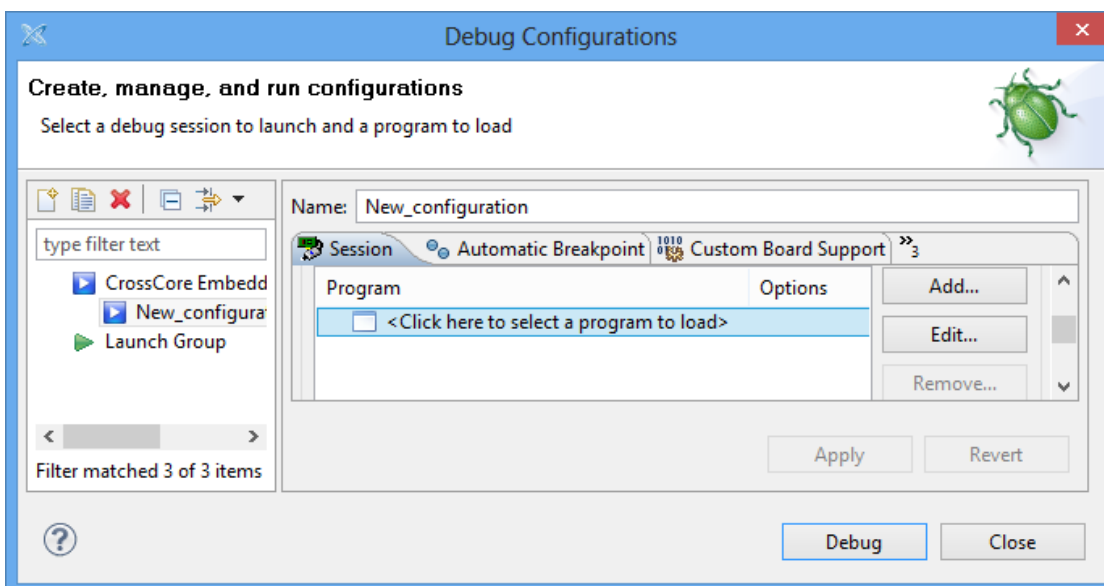


In the new window you have to select the same target configuration as in the TESSY environment, in our example it is processor "BF518", target "Simulator", and Platform "ADSP-BF5xx Single Processor Simulator". Click **Finish** when done.

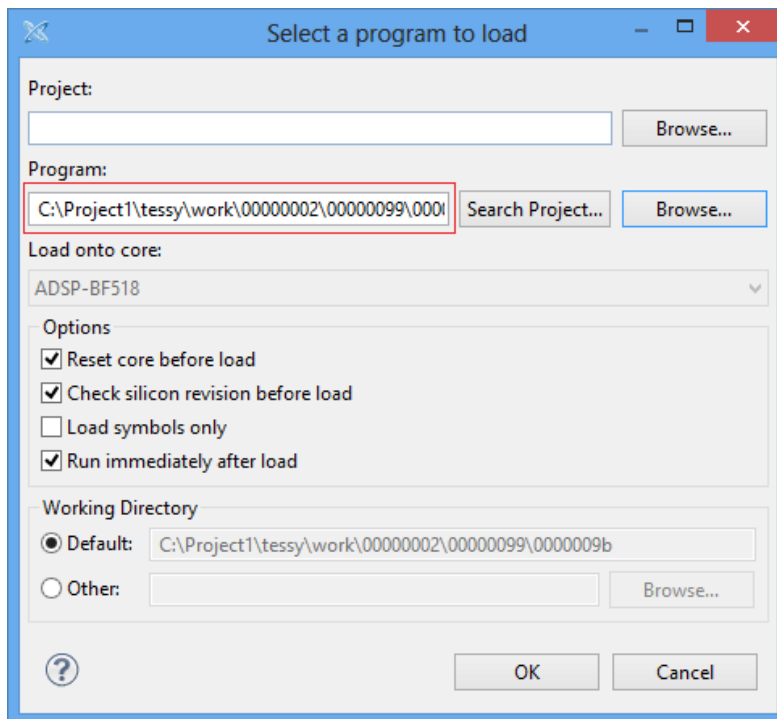




Afterwards click "<Click here to select a program to load>" within the **Session** tab of the **Debug Configurations** dialog.



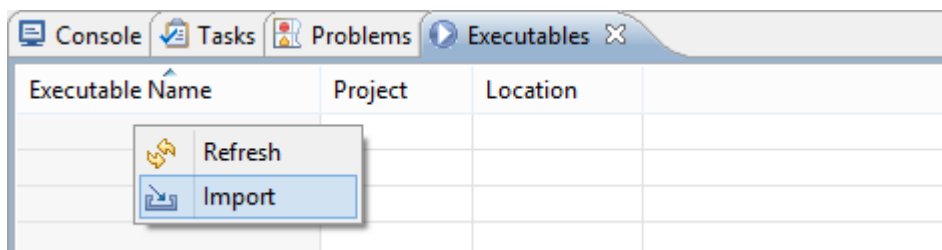
Within the **Select a program to load** dialog copy and paste the path from TESSY's Console view to the **Program:** path field and click **OK**.



Then apply the debug configurations and click **Debug**. The **Debug perspective** will open and the loaded program will stop at function main.

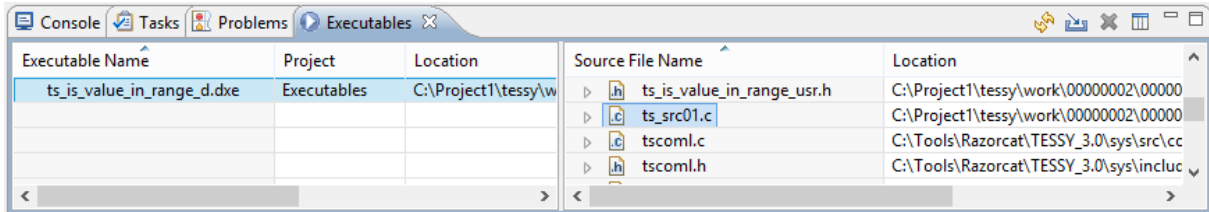
4.3 Step 3: Set a breakpoint at your test object

After loading the test driver executable into the CrossCore Embedded Studio debugger you need to set a breakpoint at your test object. Open the **Executables** view by selecting **Window -> Show view** from the menu--if it is not present. Then open the context menu anywhere within the **Executables** view and select **Import**.

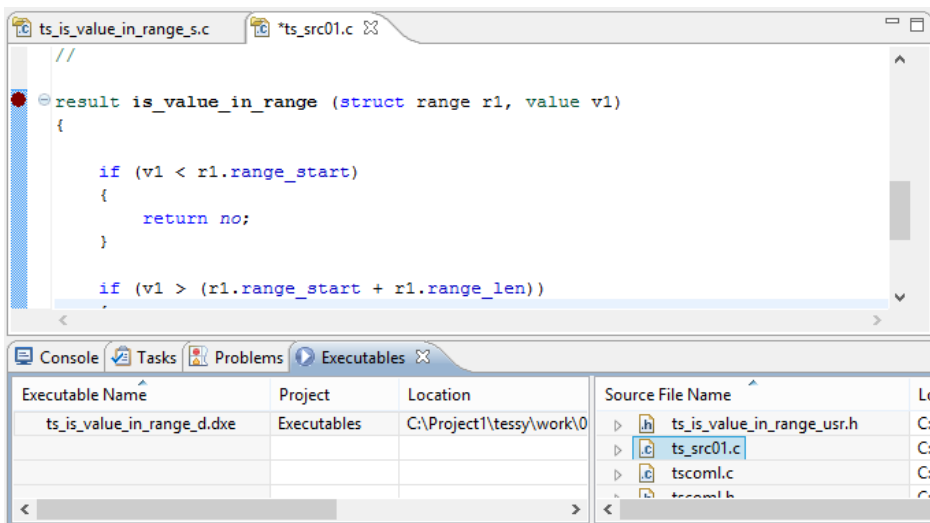


In the browser window copy and paste again the path from the TESSY Console and import the executable. After selecting the executable the relating source files will be displayed in the right part of this view.

TESSY Application Notes

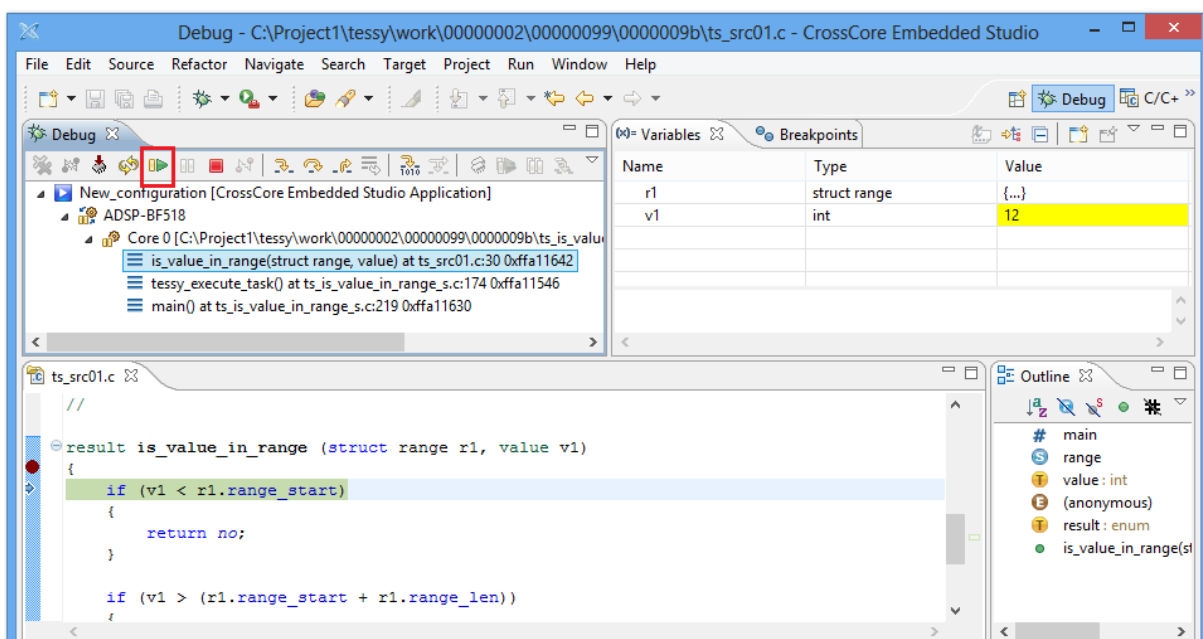


Double-click the **ts_src01.c** source file entry to open the source view. Scroll to your test object and set a breakpoint at the start of your test object function.



4.4 Step 4: Run until the test object

Now you can start the test execution within CrossCore Embedded Studio by pressing the **Resume** button and run until the breakpoint is reached.



The test data of the first (selected) test step will be available and you can debug your code. Press the **Resume** button again to run until the next test step. Pressing the **Resume** button after the last test step will result in an endless loop indicating the end of the test. You can close the debugger and go back to TESSY for further testing.

Please note: You need to reset the TEE attribute **Generate Builtin Data** to the default value (**false**) after debugging to switch back to the automatic test execution from within TESSY.

5 Known Issues

5.1 Connection problem or tthd: Java processes died unexpectedly

The Java class paths of the CrossCore Embedded Studio may change from one CrossCore Embedded Studio version to another. Therefore TESSY provides the TEE Environment Attribute **TpsdkServer** which contains the path to CrossCore Embedded Studio debugger's Java API package. Adjust the path if the TEE attribute indicates that the path was not found.

5.2 Executing test cases separately

If the test cases are executed separately you can set the wait time between the test case executions to a higher value, e.g. 2000 ms. Set the environment attribute **Separately Wait Time** to **2000**. This is especially required if you are using an emulator.